UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DISEÑO Y DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE COMANDOS BASADO EN EL USO DE BCI PARA PERSONAS CON DISFUNCIONALIDAD MOTORA Y DEL HABLA

Ryan Aiden Roncero Penistone

Tutor: José Colás Pasamontes

JULIO 2015

DISEÑO Y DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE COMANDOS BASADO EN EL USO DE BCI PARA PERSONAS CON DISFUNCIONALIDAD MOTORA Y DEL HABLA

Ryan Aiden Roncero Penistone

TUTOR: José Colás Pasamontes

Human Computer Technology Laboratory

Escuela Politécnica Superior

Universidad Autónoma de Madrid

JULIO 2015

Resumen

Este Trabajo Fin de Grado consiste en el diseño y desarrollo de un sistema de reconocimiento de comandos usando una interfaz cerebro-máquina (BCI) como dispositivo físico para leer las ondas cerebrales de una persona.

Para ello, se han diseñado dos algoritmos que permiten detectar el movimiento de ojos de un sujeto a partir de las señales cerebrales que lee el BCI.
Se han elegido tres clases distintas de movimientos oculares para clasificar, éstas son un movimiento de ojos hacía la izquierda, un movimiento de ojos hacia la derecha y un movimiento de ojos en dirección vertical (arriba o abajo).

Para poder clasificar estos tres movimientos se han escogido dos técnicas diferentes. La primera de ellas se basa en un método no paramétrico haciendo uso de umbrales para tomar una decisión, mientras que la segunda técnica consiste en el empleo de una red neuronal artificial para clasificar la dirección, incluyendo también un umbral para tomar una decisión de clasificación.

Para lograr diseñar y desarrollar estos algoritmos, se explican los pasos que se han seguido para cada uno de ellos, así como las herramientas informáticas que se han utilizado para su creación.

También se detalla el proceso llevado a cabo para la captura de las ondas cerebrales y así poder crear una base de datos a partir de la cual se calcula el número de veces que los algoritmos fallan o aciertan en su clasificación de direcciones.

Por último, se muestran una serie de tablas que contiene el porcentaje de aciertos y fallos para los dos algoritmos desarrollados y para cada uno de los sujetos, incluyéndose también dos tablas comparativas entre ambos algoritmos para poder disponer de una visión global de cada uno de ellos.

Palabras clave

Interfaz cerebro-máquina, EEG, Red Neuronal Artificial, Electrodo, Sistema 10-20, Fase de Entrenamiento, Fase de Test.

Abstract

This project consists of the design and development of a system designed to recognize commands using a brain computer interface, which is a device able to read the brain waves of a person.

To achieve this goal, two algorithms have been designed in order to detect the eye movements of different individuals by using the brain waves the brain computer interface can read. Three different types have been used for eye movement classification. These are eye movement to the right, eye movement to the left and eye movement in a vertical direction (up or down).

In order to classify these movements two techniques have been used. The first one uses a non parametric method using a threshold to make a decision, while the second technique consists in using an Artificial Neural Network to classify the directions, in addition to using a threshold to make a classification decision.

The main steps which are needed to design and develop these algorithms are explained for both cases, and also the software tools that are used to create them.

The process carried out to capture the brain waves is also explained in this project, as well as the process used to create the database, which will later be used to calculate the percentages of successes and failures of both algorithms.

Finally, a series of tables are shown, all of which contain the percentages of successes and failures of both the algorithms designed and of each one of the subjects, together with two extra tables which are used to make a comparison between both systems, therefore making it easier to have a global vision of both.

Keywords

Brain Computer Interface, EEG, Artificial Neural Network, Electrode, 10-20 system, Training Phase, Testing Phase.

Agradecimientos

Quiero comenzar este trabajo dando las gracias a mi tutor José Colás por toda la ayuda y dedicación que me ha dado para poder sacar adelante este trabajo, gracias a él he conseguido adquirir unos conocimientos que antes no poseía y he aprendido el esfuerzo que es necesario realizar para alcanzar unos objetivos tan complejos.

En segundo lugar, quería agradecer la enorme ayuda y confianza que mi familia y mis mejores amigos han depositado en mí, los cuales siempre han estado presentes en los momentos más complicados, dándome en todo momento el ánimo y apoyo necesario para superar los obstáculos con los que me he ido encontrando a lo largo de toda la carrera.

Pero sobretodo, quiero dar las gracias a mi padre y a mi madre. Gracias a ellos, he aprendido que los retos difíciles a los que hay que enfrentarse en la vida sólo se consiguen con mucha dedicación, esfuerzo y sacrificio. También quería agradecer la paciencia que han tenido conmigo a lo largo de esta importante etapa en mi vida, y la enorme confianza que me han dado para conseguir llegar a ser ingeniero.

Por último, dar de nuevo las gracias a todos los que me rodean por todos los valores que me habéis transmitido y los conocimientos que he aprendido con vosotros.

Sin la gente que quiero, nada de esto habría sido posible.

Gracias a todos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE ECUACIONES

1 Introducción

1.1 Motivación

Existen en la actualidad una gran cantidad de personas que poseen algún tipo de discapacidad y que impide que lleven a cabo una vida normal en el día a día.
Un ejemplo importante es la gente que posee tetraplejía, la cual provoca una parálisis total o parcial de las extremidades del cuerpo, que se produce fundamentalmente cuando la médula espinal es dañada seriamente.

En las últimas décadas se han desarrollado múltiples tecnologías que permiten mejorar la vida de gente con discapacidades de este tipo, intentando solventar la situación en la que se encuentran, ya que no son capaces de realizar acciones cotidianas por sí mismos, necesitando siempre la ayuda de una persona ajena.

Una importante e innovadora tecnología es el uso de interfaces hombre-máquina o también conocido como Brain Computer Interface (BCI). Este tipo de dispositivos son una especie de casco que se sitúa encima de la cabeza de una persona, y gracias al uso de sensores llamados electrodos, son capaces de leer las señales cerebrales que una persona genera.

Aunque en principio estos dispositivos pueden no parecer demasiado útiles, en los últimos años se han encontrado una gran variedad de aplicaciones importantes, y todas basándose en el hecho de que las señales cerebrales proporcionan de manera indirecta, información relativa a las expresiones faciales, movimiento de ojos, movimientos de cabeza y datos relativos a pensamientos de una persona.

Mediante el uso de este tipo de interfaces, se han investigado y desarrollado algoritmos que permiten clasificar tareas mentales, de forma que cualquier persona que posea algún tipo de discapacidad pueda usar la interfaz, y con solo pensar en una determinada acción, se pueden controlar otros objetos electrónicos, como pueden ser brazos electrónicos o piernas electrónicas.

El desarrollo de técnicas usando BCI, permitiría mejorar la vida de personas con discapacidad haciendo que su vida sea mucho más fácil y no queden excluidos socialmente.

Este trabajo se centra en el análisis de dos algoritmos distintos para detectar un movimiento de ojos a partir de las señales cerebrales que proporciona el BCI comercial *Emotiv Epoc*.

1.2 Objetivos

El objetivo principal que se persigue con este proyecto es diseñar y desarrollar un algoritmo que permita el reconocimiento del movimiento de ojos de una persona, haciendo uso de la interfaz cerebro-máquina comercial *Emotiv Epoc*.

Aunque éste es el objetivo principal, existen otros que también se intentan conseguir durante la realización de este trabajo, los cuales se nombran en los siguientes puntos:

- Entender qué es un BCI (Brain Computer Interface), como funciona, sus características básicas y las principales aplicaciones que tienen hoy en día este tipo de dispositivos.

- Conocer cuáles son las características principales que forman parte de señales electroencefalográficas.

- Conocer cuáles son los principales métodos y las técnicas que se utilizan para analizar y procesar señales electroencefalográficas.

- Aprender y usar distintos tipos de algoritmos que permitan distinguir entre un conjunto de clases, así como el funcionamiento que poseen para conseguir dicho objetivo.

- Aprender a crear una base de datos que contiene ficheros cuyos datos son señales cerebrales.

- Diseñar y desarrollar dos algoritmos distintos que permitan separar varias clases y así poder realizar una comparación entre ambos.

1.3 Organización de la memoria

Este Trabajo Fin de Grado se ha estructurado como se muestra en los siguientes puntos:

- Capitulo 1: Introducción: En este capítulo se hace una introducción hablando de los objetivos principales que se pretenden conseguir con la elaboración de este trabajo, y la motivación que ha hecho que este trabajo se lleve a cabo.

- Capitulo 2: Estado del Arte: En este capítulo se hace una revisión del estado del arte acerca de las tecnologías llamadas interfaces cerebro-máquina, explicando sus características principales, cuales son las aplicaciones que se han desarrollado con respecto a este tipo de dispositivos, nombrándose finalmente algunas empresas importantes que fabrican y venden comercialmente interfaces cerebro-máquina.

- Capitulo 3: Diseño: En esta sección se explica cual es el objetivo principal que se pretende conseguir con este trabajo, y se habla también de todos los medios que se han utilizado, tanto de hardware como de software, para conseguir dicho objetivo. Por último se habla del proceso que se ha seguido para crear una base de datos que ha permitido trabajar con las señales características de interfaces cerebro-máquina, la cual ha permitido estudiar este tipo de señales para así poder posteriormente hallar un conjunto de estadísticas.

- Capitulo 4: Desarrollo: En este apartado, se explica todo el proceso que se ha llevado a cabo para conseguir el objetivo principal de este trabajo, y para ello se desarrolla con mucho detalle el algoritmo usando dos aproximaciones distintas para conseguir la meta propuesta.

- Capitulo 5: Integración, pruebas y resultados: Se muestran una serie de tablas en las cuales se pueden ver los resultados obtenidos para ambos algoritmos desarrollados, así como dos tablas comparativas entre los dos métodos.

- Capitulo 6: Conclusiones y trabajo futuro: Esta última sección explica las principales conclusiones a las que se ha llegado a partir del diseño y desarrollo de los algoritmos creados. Por último, se habla del futuro trabajo que se podría llevar a cabo con el fin de continuar investigando en aplicaciones relacionadas con interfaces cerebro-máquina.

2 Estado del arte

2.1 Brain Computer Interface.

Una interfaz cerebro-computador, también conocido como BCI (Brain Computer Interface), es un dispositivo que capta las señales cerebrales de una persona para posteriormente enviarlas a un ordenador las cuales serán procesadas.

Estas ondas cerebrales que son conocidas como señales electroencefalográficas (EEG) proporcionan de manera indirecta información relativa a funciones del cerebro, como pueden ser tareas mentales, acciones motoras, o expresiones faciales entre otras.

Existen cinco tipos de ondas que se distinguen por el rango de frecuencias que abarcan: Las ondas Delta se encuentran en el rango que oscila entre los 0.5 y los 4 Hz, las ondas Theta pertenecen al rango de 4 a 7 Hz mientras que las ondas Alpha abarcan desde los 8 hasta los 13 Hz, las ondas Beta se encuentran en el rango de 13 a 30 Hz y las ondas Ganma oscilan entre los 35 Hz y frecuencias superiores. Cada una de estos tipos de señales está relacionada con una tarea concreta, ya que por ejemplo las ondas Beta proporcionan información acerca de tareas mentales, atención o tareas en las cuales se piensa de forma activa.
Otro tipo de propiedad que poseen las ondas cerebrales es que la amplitud de la actividad eléctrica registrada en los sensores del BCI rondan los microvoltios, comparado con la actividad eléctrica que genera una neurona, la cual se encuentra entorno a los milivoltios, esto se debe fundamentalmente al hecho de que este tipo de ondas se filtran y atenúan a medida que van atravesando las diferentes capas del cerebro y el cráneo.

2.2 Aplicaciones utilizando BCI

Durante muchos años, una de las principales líneas de investigación acerca de este tipo de aparatos ha sido básicamente el procesado de las señales EEG para posteriormente poder extraer una serie de características (features) y así poder clasificarlas con distintos algoritmos de clasificación. Estos métodos permitirían transformar las señales EEG en algún tipo de comando para así mejorar la vida de personas que posean algún tipo de discapacidad motora o del habla, como por ejemplo personas tetrapléjicas.

Este proceso se muestra de forma esquemática en la figura 1:

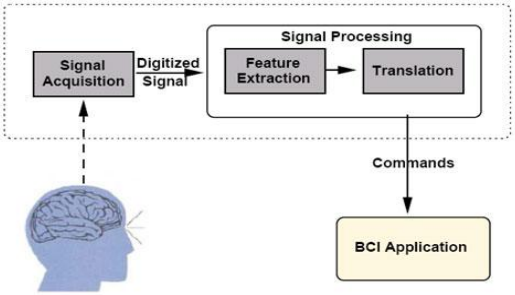

Figura 1. Esquema típico para aplicaciones BCI.

Este trabajo, se ha centrado en desarrollar un algoritmo que permite clasificar la dirección en la cual un sujeto mueve los ojos (derecha, izquierda, arriba o abajo), a partir de las señales cerebrales que nos proporciona el BCI. Para ello se ha llevado a cabo todo el procesamiento necesario y la posterior extracción de características para clasificar dicho movimiento.

De acuerdo al proceso de la figura 1, durante los últimos años se han investigado y desarrollado diferentes técnicas que permiten analizar y procesar las señales EEG que se obtienen gracias a interfaces cerebro-máquina, y a partir de dicho procesamiento poder extraer algún tipo de información que proporcionen este tipo de señales. Mediante la extracción de la información que puedan proveer las ondas cerebrales, ésta puede ser usada para controlar dispositivos electrónicos y así facilitar la vida a una gran cantidad de personas.

Entre las aplicaciones que se han conseguido en los últimos años, destaca principalmente la creación de algoritmos que permiten controlar brazos electrónicos utilizando solo el pensamiento y sin tener que llevar a cabo ningún tipo de movimiento físico. Otras tecnologías que se han conseguido desarrollar corresponden al movimiento del cursor de un ordenador con tan solo pensar en la dirección en la cual se quiere mover dicho cursor.

Estos son sólo algunos de los métodos que se han desarrollado en las últimas décadas para crear aplicaciones de BCI.

2.3 Técnicas de BCI para clasificaciones.

A lo largo de las últimas décadas, se han realizado multitud de experimentos y estudios que han servido para desarrollar algoritmos y métodos capaces de clasificar la información procedente de señales EEG.
Este apartado habla sobre las principales técnicas que se usan hoy en día para la clasificación de información que puedan proveer las señales cerebrales captadas con un BCI.

Entre las principales técnicas que se usan, a continuación se nombran las más utilizadas:

- Redes Bayesianas: Se trata de un método estadístico que relaciona un conjunto de variables aleatorias usando un grafo. Utiliza el teorema de Bayes como algoritmo, y es un algoritmo de tipo paramétrico, esto quiere decir que va aprendiendo nuevos conocimientos a partir de los ejemplos que ha recibido previamente.

- Redes Neuronales Artificiales: Las redes neuronales artificiales (Artificial Neural Network) están formadas por un conjunto de nodos los cuales se llaman neuronas y simulan las neuronas del cerebro. Estas neuronas tienen una serie de pesos asignados formando una función de transferencia, por lo que la asignación de estos pesos es lo que da lugar a una salida u otra dependiendo de los datos de entrada. Es decir, actúa como una caja negra en la cual las salidas de la red dependen de los datos de entrada y de la función de transferencia que se forme mediante los pesos de las neuronas.

2.4 BCI Comerciales

En esta subsección se nombran las principales empresas que venden interfaces cerebro-máquina de forma comercial, ya que hasta hace pocos años este tipo de dispositivos solo eran usados en centros de investigación y universidades.

Entre las empresas que venden BCI destacan:

- Neurosky: Esta empresa fundada en San José, California, vende un BCI llamado *Mindset*, el cual consiste en un circuito integrado que lleva a cabo lecturas de señales cerebrales haciendo uso de solo un canal EEG que se posiciona sobre la frente. La información que se captura con el canal se recoge gracias a un electrodo el cual amplifica, filtra y entrega dos valores, uno de ellos corresponde a un parámetro de atención de una persona mientras que el otro parámetro que se entrega corresponde a un dato de relajación.
  Gracias a estos dos parámetros se han desarrollado aplicaciones y juegos para mejorar la capacidad de atención y la medición de relajación.
  Los datos se envían usando tecnología Bluetooth entre el interfaz y un ordenador, y el BCI tiene un precio de 200 dólares en Estados Unidos.

- Ocz Technologies: Se trata de otra empresa también originaria de San José, la cual fue una de las primeras que vendió un BCI comercial. Su producto principal es la interfaz cerebro-máquina llamado *Neural Impulse Actuator*, el cual está formado por tres electrodos que se sitúan sobre la frente de una persona, uno de ellos lee señales electrooculográficas, otro lee señales electroencefalográficas y el ultimo lee señales electromiográficas. Esta interfaz ha sido utilizado principalmente como joystick para el uso de videojuegos y se conecta mediante un cable usb al ordenador, por lo que no es una forma cómoda de utilizar el BCI.

- Emotiv Epoc: Es una empresa australiana fundada en 2003 por varios científicos y ejecutivos. Después de varios años de investigación en colaboración con universidades y centros de investigación, se desarrolló el BCI comercial *Emotiv Epoc*, el cual es una interfaz cerebro-máquina que contiene un total de 14 electrodos mas 2 de referencia y es capaz de captar múltiples tipos de señales. Incorpora un testbench con su compra y existen multitud de aplicaciones gratuitas y de pago que se pueden adquirir en la página web de la empresa. Este ha sido el BCI que se ha utilizado para este trabajo, y sus características se explican de forma detallada en la sección de diseño de este trabajo.

3 Diseño

3.1 Diseño de un sistema de detección de movimiento de ojos a partir de señales EEG.

En este apartado se explica el análisis y diseño de un sistema que se basa en el reconocimiento del movimiento de ojos de un sujeto a partir de las señales EEG que entrega el BCI, con un determinado porcentaje de acierto y fallo.
También se explica el hardware y el software utilizado para el desarrollo de este sistema, así como las características principales que estos poseen y la manera por la cual funciona.

3.2 Interfaz cerebro-máquina utilizado: Emotiv Epoc

Para la adquisición de las señales EEG que se han capturado en este trabajo se ha hecho uso del BCI comercial llamado *Emotiv Epoc*. Esta interfaz cerebro-máquina está formada por 14 sensores más 2 de referencia, cada uno de las cuales capta las señales electroencefalográficas en diferentes puntos del cuero cabelludo de una persona, de acuerdo al sistema internacional 10-20.
El sistema 10-20 es una nomenclatura estándar e internacional la cual se usa para describir la situación de los sensores del BCI encima del cuero cabelludo de una persona. Básicamente este sistema especifica que sensores adyacentes de una interfaz cerebro-máquina están separados una distancia entre el 10 ó 20% de la distancia existente entre el extremo trasero y frontal o el extremo izquierdo y derecho del cuero cabelludo.
Cada uno de estos sensores está nombrado con una referencia única dependiendo de la posición de estos sobre el cuero cabelludo. Cada referencia está formada por una letra y un número. La letra especifica cuál es el lóbulo sobre el que se sitúa el sensor mientras que el número designa la localización del sensor sobre el hemisferio. Las letras F, T, C, P y O se refieren a los lóbulos frontales, temporales, centrales, parietales y occipitales respectivamente. Los números pares 2, 4, 6 y 8 se relacionan con los electrodos situados en el hemisferio derecho del cerebro mientras que los números impares 1, 3, 5 y 7 se refieren a los electrodos situados en el hemisferio izquierdo del cerebro.

La figura 2 muestra la posición de los electrodos para una interfaz cerebro-máquina que contiene un total de 64 electrodos, haciendo uso del sistema 10-20:

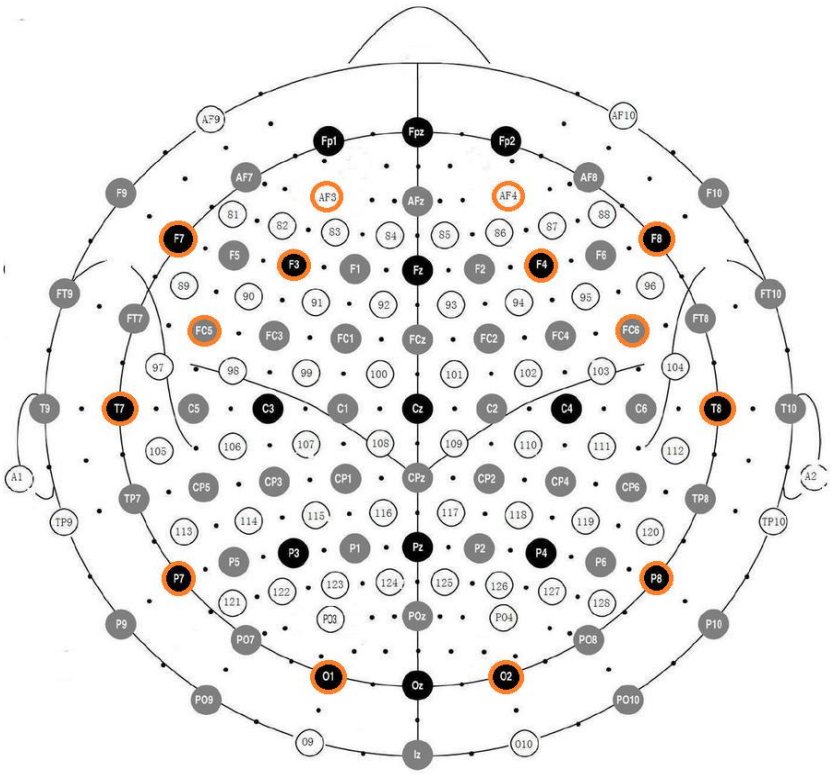

Figura 2. Distribución de electrodos de BCI con 64 electrodos - Sistema 10-20.

Sin embargo, no todas las interfaces poseen tantos electrodos como la interfaz especificada en la figura 2. Para este trabajo como ya se mencionó, se ha usado *Emotiv Epoc*, un BCI con un total de 14 sensores más 2 de referencia, y cuyo sistema 10-20 se puede visualizar en la figura 3. Para poder hacerse una idea de los sensores que usa *Emotiv Epoc* en comparación con otras interfaces que poseen un mayor número de electrodos, la figura 2 muestra los sensores que usa *Emotiv Epoc*, los cuales están rodeados con un círculo naranja.

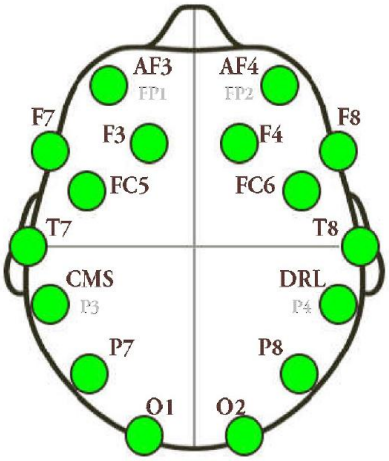

Figura 3. Distribución de electrodos de Emotiv Epoc.

Entre las características técnicas del BCI que se ha usado para este trabajo destacan la frecuencia de muestreo la cual tiene un valor de 128 Hz, una resolución de 14 bits para las señales capturadas mediante los electrodos, y un ancho de banda que oscila entre 0.2 y 45 Hz, y posee un rango dinámico cuyo valor son 8400 μV . También hace uso de filtros *Notch* digitales a 50 y 60 Hz. Estos filtros sirven para suprimir el ruido que genera la corriente de alimentación del BCI y que se puede acoplar a las señales captadas con los electrodos. La duración típica de la batería se encuentra alrededor de una duración de 12 horas aproximadamente.
La figura 4 es una fotografía de la interfaz que se ha usado para este trabajo (*Emotiv Epoc*):

Figura 4. Emotiv Epoc.

Para más información acerca del BCI Emotiv Epoc, se puede encontrar en su página web oficial en el siguiente link:

https://emotiv.com/epoc.php

3.3 Software usado: TestBench

Como software principal para la adquisición de las señales EEG se ha usado un *TestBench* el cual proporciona una amplia variedad de aplicaciones.
Entre las principales funciones que este software trae consigo destacan la posibilidad de visualizar, guardar y reproducir las señales EEG captadas por el BCI, así como su visualización tanto en el dominio del tiempo como en dominio de la frecuencia, permitiendo también elegir el tipo de ventana a utilizar (*Hanning, Hamming, Hann, Blackman o Rectangular*).

Otra característica que trae consigo el *TestBench* es la visualización de la potencia en las distintas bandas de frecuencia (*Delta, Theta, Alfa, Beta y Custom*), y también se muestra un diagrama de la calidad de contacto de los sensores con el cuero cabelludo, para así poder asegurar que se captan las señales EEG con una calidad razonable.

También se pueden visualizar las señales que proporciona el giroscopio del BCI, y por último se puede ver información relativa al envió de paquetes entre la interfaz y el ordenador, la cual sirve para asegurar que toda la información procedente de la interfaz ha llegado de forma correcta al ordenador de destino.

La figura 5 representa una captura del *TestBench*, en el cual se pueden visualizar las características anteriormente explicadas:

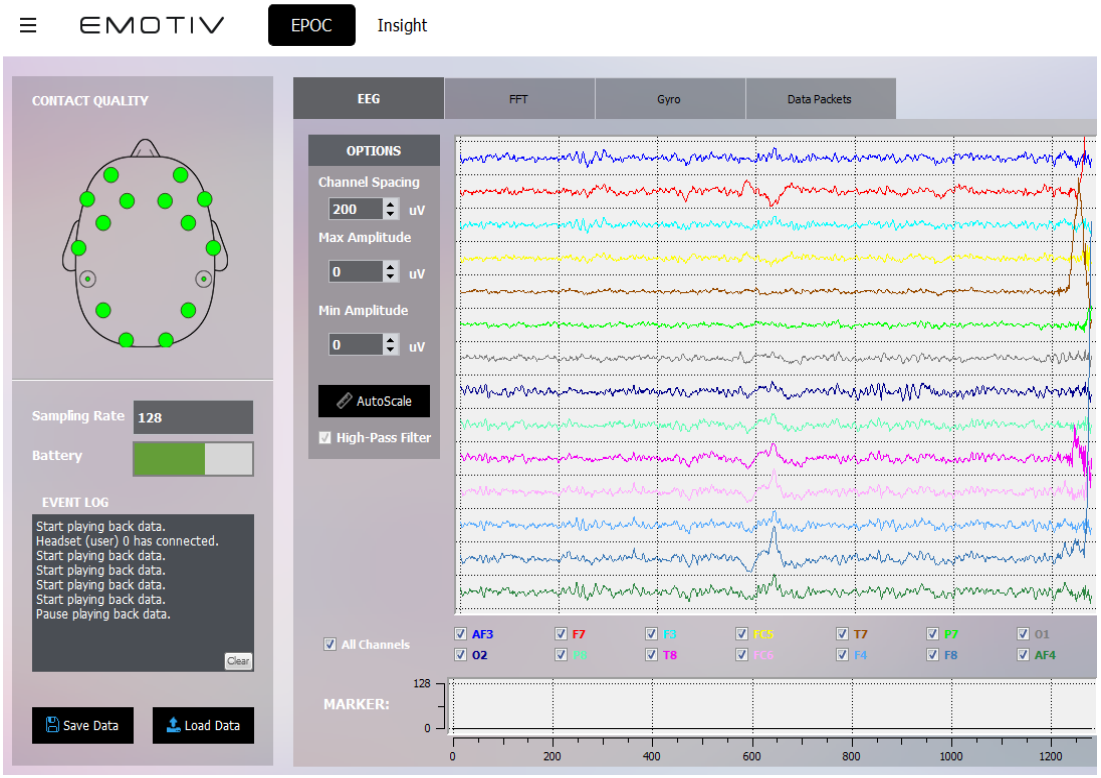

Figura 5. Testbench de Emotiv Epoc.

3.4 Matlab

Para este trabajo se ha hecho uso de *Matlab* como software principal para realizar todo el procesamiento y análisis de las señales electroencefalograficas obtenidas con el BCI. También se ha utilizado *Matlab* para escribir el código necesario relativo a los datos y estadísticas que se han tenido que calcular.

3.5 EEGLAB

Debido principalmente a que la interfaz cerebro-máquina que se ha utilizado en este trabajo almacena la captura de señales electroencefalográficas en ficheros con extensión *edf*, ha sido necesario para poder leer dichos ficheros el uso de un Toolbox de *Matlab* llamado *EEGLAB*, el cual permite leer este tipo de archivos, además de disponer de una gran cantidad de herramientas diseñadas para procesar este tipo de señales.

EDF son las siglas para *European Data Format*. Se trata de un formato de archivos usado principalmente para intercambiar y guardar señales biológicas y médicas. Es por esto que las señales electroencefalográficas al ser señales médicas, se almacenan en archivos con extensión *edf*.

Debido a esto, se ha usado *EEGLAB* como Toolbox de *Matlab* necesario para leer dichas señales y así poder importarlas para que puedan ser procesadas posteriormente.
Una vez instalado EEGLAB, para poder ejecutarlo solo hay que introducir el comando *eeglab* en la consola que proporciona *Matlab*.

3.6 Neural Network Toolbox

Otra herramienta que ha sido necesaria para este trabajo ha sido el uso de otro Toolbox que proporciona *Matlab* llamado *Neural Network Toolbox*, el cual proporciona funciones y aplicaciones para diseñar, construir, entrenar y simular redes neuronales artificiales haciendo uso de distintos métodos y algoritmos.
También se puede usar *Neural Network Toolbox* para aplicaciones como pueden ser el reconocimiento de patrones, métodos de clustering, data fitting o el modelado y control de sistemas dinámicos entre otros.
Para este trabajo *Neural Network Toolbox* se ha utilizado para el diseño, construcción, entrenamiento y simulación de una red neuronal artificial que permita distinguir entre tres clases diferentes.
El método que se ha seguido para su diseño y construcción se explica más adelante de una manera bastante detallada.
Para más información acerca de *Neural Network Toolbox*, se puede visitar el siguiente link:

http://es.mathworks.com/products/neural-network/

3.7 Captura de las señales EEG y creación de la base de datos

En este apartado se explica el método que se ha llevado a cabo para la captura de las señales EEG de distintos usuarios con el fin de crear una base de datos que sirva más adelante para analizar y procesar dichas señales.

Como ya se ha comentado, el objetivo de este trabajo es crear un sistema que permita el reconocimiento del movimiento de ojos de un sujeto a partir de las señales EEG. Para la captura de esta base de datos se ha hecho uso del BCI *Emotiv Epoc* como dispositivo físico para la captura de dichas señales, el cual envía los datos a un ordenador usando tecnología Bluetooth para que puedan ser leídos y guardados gracias al uso del *TestBench* anteriormente explicado.

Para poder crear la base de datos, el método a seguir ha sido el mismo para todos los usuarios a los que se ha realizado el experimento. A continuación se explican los pasos que se han llevado a cabo:

- En primer lugar es necesario humedecer los sensores del BCI con agua salina con el fin de crear una conexión entre el cuero cabelludo y el dispositivo. Existen interfaces hoy en día para las cuales no es necesario llevar a cabo este proceso, pero para el interfaz *Emotiv Epoc* este paso es un requisito imprescindible para poder establecer una conexión. *Emotiv Epoc* trae consigo por defecto un pequeño bote de agua salina, no obstante si se diese el caso de que el bote se quedase sin agua salina, éste se puede conseguir en cualquier farmacia.

- Una vez realizado el paso anterior, se procede a situar el casco por encima del cuero cabelludo de la persona a la cual se vayan a captar las señales cerebrales, asegurando siempre que existe una buena conexión de la interfaz cerebro-máquina con la cabeza, la cual se puede comprobar gracias al esquema de la calidad de contacto de los sensores con el cuero cabelludo que nos proporciona el *TestBench (ver figura 5)*. En el caso de que no exista una calidad aceptable de conexión, el método a seguir es incrementar la humedad en los sensores hasta que se consiga una buena conexión.

- Cuando este proceso se haya conseguido, el siguiente paso es proceder a la captura de las señales para que puedan ser almacenadas en la base de datos.
  Para empezar, se les pidió a los usuarios que estableciesen un modo de relajación para que se pudiese empezar a grabar las señales. Una vez que los usuarios estuvieran relajados se empezaba la grabación de las ondas cerebrales mediante el uso del *Testbench*.
  Como el objetivo principal del trabajo ha sido la captura de las señales cuando el sujeto hiciese un movimiento de ojos, se decidió emitir un pitido cuando pasasen cinco segundos desde el comienzo de la grabación, este pitido era la señal que indicaba que el usuario tenía que mover los ojos hacia un lado (derecha, izquierda, arriba o abajo) y posteriormente volver a mirar a una posición central durante el tiempo restante, pasando de nuevo a un modo de relajación.

Estos pasos son los que han formado parte del proceso a seguir para todos los usuarios, procurando siempre que existiese una calidad aceptable de conexión con el cuero

cabelludo y disponiendo de un lugar tranquilo en el cual no existan señales que puedan interferir de forma excesiva con las capturadas a partir de los electrodos del BCI.

Para la base de datos se escogieron un total de 5 usuarios de ambos géneros y con edades muy diversas, en gran parte para así poder comprobar si los datos cambiaban significativamente de un usuario a otro.
El número de muestras por usuario (entendiendo como muestra una grabación completa de diez segundos de duración), ha sido de 50 por clase, de las cuales se han usado la mitad como datos de entrenamiento y la mitad restante como datos de test, por lo que en total se capturaron 200 muestras por usuario.
Las clases involucradas han sido el movimiento ocular en tres direcciones, las cuales han sido movimiento de los ojos hacia la izquierda, derecha y una dirección vertical (arriba o abajo).

Una vez acabado el proceso de la captura de las señales para todos los usuarios, ya se tiene lista la base de datos que posteriormente se usará para procesar los datos.

Como ya se explicó anteriormente, cada una de las grabaciones de las ondas EEG se almacenan en archivos con extensión *edf*, por lo que para poder leer dichas señales e importarlas a *Matlab*, se usó el Toolbox *EEGLAB*, el cual permite leer y procesar este tipo de ficheros.

4 Desarrollo

En este apartado, se detalla todo el proceso que se ha seguido para poder clasificar movimientos oculares. Para dicha clasificación se han optado por desarrollar dos tipos de métodos, el primero de ellos se basa en el uso de umbrales, mientras que el segundo usa un método más estadístico, empleando una red neuronal para clasificar las direcciones.

4.1 Algoritmo de reconocimiento de movimientos oculares usando métodos no paramétricos y decisión basada en umbrales.

En esta sección se explica el método que se ha seguido para desarrollar un algoritmo que permita clasificar la dirección en la cual un usuario mueve los ojos, a partir de las señales EEG que captura el BCI, usando un método basado en umbrales.
Para ello es necesario seguir una serie de pasos, explicándose con detalle en los siguientes puntos:

4.1.1 Pre-procesamiento de las señales EEG.

En primer lugar, como ocurre con cualquier otro tipo de señal, será necesario hacer un pre-procesamiento de las señales cerebrales con el fin de eliminar las características que no forman parte de su naturaleza. Para conseguir este objetivo, se explican a continuación los pasos que se han seguido en este trabajo:

- Para empezar, una característica importante de señales EEG está relacionada con el ancho de banda que ocupan. Para señales cerebrales las frecuencias que comprenden se encuentran en el rango de 0.5 a 100 Hz.
  Aunque el ancho de banda abarca el intervalo mencionado, un significativo porcentaje de potencia se encuentra en el rango que oscila entre 0.5 y 60 Hz, es por ello que el primer paso a seguir en la etapa de pre-procesamiento se basa en filtrar las señales con un filtro paso bajo el cual tiene una frecuencia de corte de 60 Hz y un ancho de banda de 0.5 a 60 Hz. Gracias a este filtro, conseguimos eliminar ruido y frecuencias que no forman parte de este tipo de señales.

  En este trabajo, como filtro paso bajo se ha utilizado un filtro *Butterworth* de orden 8 y con una frecuencia de corte de 60 Hz tal y como se ha mencionado anteriormente.
  La principal explicación por la que usamos un filtro *Butterworth* se debe principalmente a que su respuesta en frecuencia no contiene ningún rizado y se caracteriza por ser plana tanto en la banda de paso como en la banda de rechazo, lo que hace que este tipo de filtros sean ideales para señales electroencefalográficas.

  Para conseguir este filtrado se ha hecho uso de la función *butter* que proporciona *Matlab,* para la cual es necesario pasarle como parámetros de entrada el orden del filtro y la frecuencia de corte. Los parámetros que devuelve son los

coeficientes del numerador y denominador de la función de transferencia que ha calculado la función *butter*.
Para posteriormente filtrar la señal deseada se ha utilizado la función *filter*. Ésta recibe como parámetros de entrada los coeficientes devueltos por la función *butter*, tanto de numerador como denominador, y la señal que se quiere filtrar, devolviendo como parámetro de salida la señal filtrada, en este caso la filtrada por el filtro *Butterworth*.

- También es necesario también tener en cuenta que las señales capturadas mediante interfaces cerebro-máquina llevan acopladas el ruido procedente de la red eléctrica que alimenta la interfaz, por lo que suele ser común filtrar las señales con un filtro *Notch*, cuya frecuencia central depende de la frecuencia de la red eléctrica en un determinado país. Para la mayoría de los países esta frecuencia tiene un valor de 50 o 60 Hz.
  Sin embargo, aunque este paso es estándar en el procesamiento de ondas cerebrales, la interfaz cerebro-maquina que se ha usado en este trabajo (*Emotiv Epoc*) elimina por defecto el ruido de la red eléctrica, por lo que para este trabajo este paso se puede obviar en la etapa de pre-procesamiento.

- Otro paso que hay que realizar en el procesamiento previo de ondas cerebrales se basa en eliminar la componente media de la señal, para que la señal tenga un valor medio nulo en el rango para el cual se esté haciendo dicho procesamiento. Con esta técnica conseguimos disponer de una señal que más tarde nos permitirá discriminar mejor unas ciertas características, como puede ser por ejemplo la detección de picos positivos y negativos.

- Las señales electroencefalográficas son por naturaleza muy ruidosas, ya que cualquier actividad muscular de la cara o cualquier mínimo movimiento genera una gran cantidad de ruido, por lo que siempre es necesario suavizar dichas señales.

  Un procedimiento típico consiste en aproximar las señales obtenidas con los electrodos a un polinomio de orden alto haciendo uso del método de los mínimos cuadrados. Este método busca un polinomio de un determinado orden cuyo error con respecto a la onda original usando el método de los mínimos cuadrados (*least square method*) sea el mínimo posible.
  Para este trabajo se usó un polinomio de orden 10 para las ondas obtenidas, y gracias a este método se elimina una gran cantidad de ruido al mismo tiempo que se produce un suavizado de las señales, el cual siempre es favorable a la hora de analizarlas.

  Para entender mejor este paso, llamaremos $X(t)$ a la señal que queremos suavizar y $P(t)$ será la señal suavizada, la cual es un polinomio de orden 10 el que se ha utilizado en este trabajo. $E(t)$ es la función error la cual se expresa mediante la fórmula de la ecuación 1:

$$E(t) = \sqrt{\frac{1}{b-a}\int_a^b (X(t) - P(t))^2 \, dx}$$

Ecuación 1. Función error del método de mínimos cuadrados (least square method).

Este algoritmo trata de encontrar los coeficientes del polinomio $P(t)$ de orden 10 para los cuales la función error $E(t)$ es mínima.

Estos polinomios se pueden calcular gracias a la función *polyfit* que nos proporciona *Matlab*, para la cual solo hace falta introducir como parámetros de entrada el intervalo en el cual se esté hallando la aproximación, la señal que se quiere suavizar y el orden del polinomio. Esta función devolverá a su salida los coeficientes del polinomio que se han hallado, intentando minimizar siempre la función error. (ecuación 1).

Para finalmente poder hallar el polinomio $P(t)$, se he utilizado la función *polyval*, para la cual recibe como parámetros de entrada los coeficientes devueltos por la función *polyfit* y el intervalo en el cual se quiere evaluar dicho polinomio. Su parámetro de salida corresponde finalmente a la señal suavizada $P(t)$.

En resumen, todos los pasos anteriormente explicados constituyen el pre-procesamiento estándar que es necesario llevar a cabo para señales EEG. Hay que tener en cuenta que este pre-procesamiento se aplica a todas las señales que la interfaz cerebro-maquina capte mediante sus sensores, en nuestro caso la interfaz dispone de 14 sensores por lo que será necesario aplicar el procesamiento a las 14 señales, independientemente del posterior uso que se pueda hacer de ellas.

El siguiente paso que es necesario llevar a cabo es identificar solo las señales que aporten información del movimiento de ojos, ya que los electrodos de la interfaz dependiendo de su situación sobre el cuero cabelludo captan un determinado tipo de información u otro.

Para este trabajo al querer detectar un movimiento ocular, se observó gracias al *Testbench,* que los electrodos que más información aportaban para dicho propósito eran los electrodos situados en la zona cercana a la frente y sobre ella, esto es los electrodos *AF3,AF4,F3,F4,F7,F8,F5* y *FC6*.

Principalmente se capta más información relativa al movimiento de ojos con estos sensores, debido al hecho de que están situados en una zona más cercana a los ojos y por tanto detectan mejor los movimientos oculares en comparación con el resto de los electrodos.

Por tanto, a partir de ahora se usarán solo estas ocho señales para el análisis que más adelante se hará de ellas.

En las figuras 6 y 7 se puede observar las 8 señales que se han usado para detectar un movimiento de ojos, antes de ser procesadas y previamente filtradas.

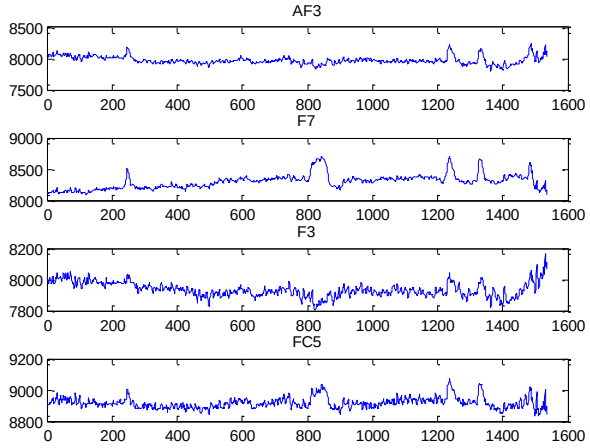

Figura 6. Señales EEG del hemisferio izquierdo.

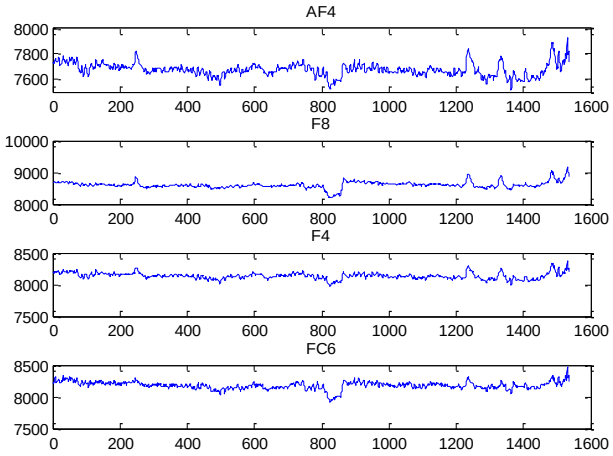

Figura 7. Señales EEG del hemisferio derecho.

Las señales *AF3, F7, F3* y *FC5* (figura 6) corresponden a los electrodos situados en la zona izquierda de la frente mientras que las señales *AF4, F8, F4* y *FC6* (figura 7) son los electrodos situados en la zona derecha de la frente. El eje X corresponde al tiempo transcurrido de las grabaciones, mostrado en número de muestras tomadas y el eje Y corresponde a la amplitud de las señales electroencefalográficas en el orden de microvoltios.

Como se puede observar en las figuras 6 y 7, las señales tienen una duración aproximada de 10 segundos, sin embargo para este trabajo solo queremos obtener el intervalo de las señales en el cual se produce un movimiento de ojos. Como se ha explicado anteriormente, durante la realización de la base de datos se decidió emitir un pitido cuando se llegasen a cinco segundos de duración desde el comienzo de la grabación. Teniendo en cuenta esto, se ha aplicado una ventana rectangular para cada una de las señales entorno a los cinco segundos y con una duración de un segundo. Con

la aplicación de esta ventana a las señales, se asegura que se selecciona el rango suficiente en el cual se produce un movimiento ocular.

Por tanto a partir de ahora, se tendrán señales suavizadas con una duración más corta que la grabación completa, pero siempre conteniendo información relativa al movimiento de ojos de un usuario determinado.

Todos los pasos que se han explicado hasta el momento forman parte del pre-procesamiento inicial que es necesario realizar, para así eliminar ruido, suavizar las señales e identificar aquellas que interesan para este trabajo.

Salvo el paso de identificación de las señales que aporten información relativa a movimientos oculares, la mayor parte de los pasos previos son los que se llevan a cabo en el procesamiento previo de señales cerebrales, sea cual sea el objetivo final que se quiera conseguir.

4.1.2 Extraccion de features.

En este apartado se explica cuales han sido los features o características necesarias que se han tenido que extraer con el fin de poder clasificar las señales según la dirección de movimiento de ojos de un sujeto.

Para ello hay que tener en cuenta una característica importante que poseen las señales que se han seleccionado previamente. Aunque ya se ha explicado el motivo por el cual se han elegido los sensores situados en la zona frontal de la cabeza, se puede observar con cuidadosa atención una característica que se cumple para los sensores elegidos. Se observó principalmente que los electrodos *AF3, F7, F3* y *FC5* producían una polaridad opuesta a los electrodos *AF4, F8, F4* y *FC6* en el momento en el cual se producía un movimiento de ojos. Esta característica va a ser muy importante y determinante para poder extraer uno de los principales features que se ha usado en este trabajo.
Gracias a este hecho, se ha usado una señal que se forma mediante una simple combinación aritmética de los ocho sensores elegidos, esta señal la llamaremos $X\_Auxiliar$, la cual se forma con la fórmula de la ecuación 2:

$$X\_Auxiliar = AF3 + F3 + F7 + FC5 - (AF4 + F4 + F8 + FC6)$$

Ecuación 2. Señal auxiliar usada para discriminar entre clases.

Hay que tener en cuenta que las señales que se han usado en esta combinación corresponden a las señales que han sido suavizadas y pre-procesadas en su correspondiente etapa, es decir no se tratan de aquellas ondas obtenidas directamente con los electrodos de la interfaz.
Esta señal nueva es lo suficientemente discriminante para poder distinguir entre las direcciones izquierda, derecha y una dirección vertical (arriba o abajo), y todo ello gracias al simple hecho de que los electrodos situados en la zona izquierda de la frente producen polaridades inversas a los situados en la zona derecha.

Otro feature que ha sido necesario calcular corresponde al área bajo la curva de la señal auxiliar que se acaba de hallar.
Esta área ha sido calculada mediante el método del trapezoide, el cual funciona principalmente aproximando el área bajo la curva a un trapezoide y posteriormente se calcula la superficie correspondiente.
El método del trapezoide se halla usando la fórmula que muestra la ecuación 3:

$$\int_a^b X(t)dt = \frac{1}{2}(b-a)(X(a)+X(b))$$

Ecuación 3. Área bajo la curva de una señal usando el método del trapezoide.

En la fórmula de la ecuación 3, $X(t)$ corresponde a la señal para la cual estamos calculando su área, mientras que *a* y *b* corresponden a los límites del intervalo para el cual se está hallando dicha superficie.
La función que se ha usado en este trabajo para hallar el área ha sido *trapz*, la cual es una función que nos proporciona *Matlab* cuyo parámetro único de entrada es la señal para la cual queremos calcular su área. El parámetro de salida de esta función corresponde al área hallado usando el método del trapezoide.

En resumen, los dos features necesarios que se han necesitado hallar para poder distinguir entre las direcciones de los movimientos oculares son una señal auxiliar que máxima el margen entre las direcciones, y un área que nos permitirá también discriminar entre los cuatro movimientos.

Finalmente se usa el área bajo la curva para discriminar y hallar los umbrales necesarios para diferenciar entre las cuatro direcciones.
En la figura 8 se puede visualizar las distribuciones de las áreas para las cuatro direcciones de movimientos oculares para un total de 25 muestras:

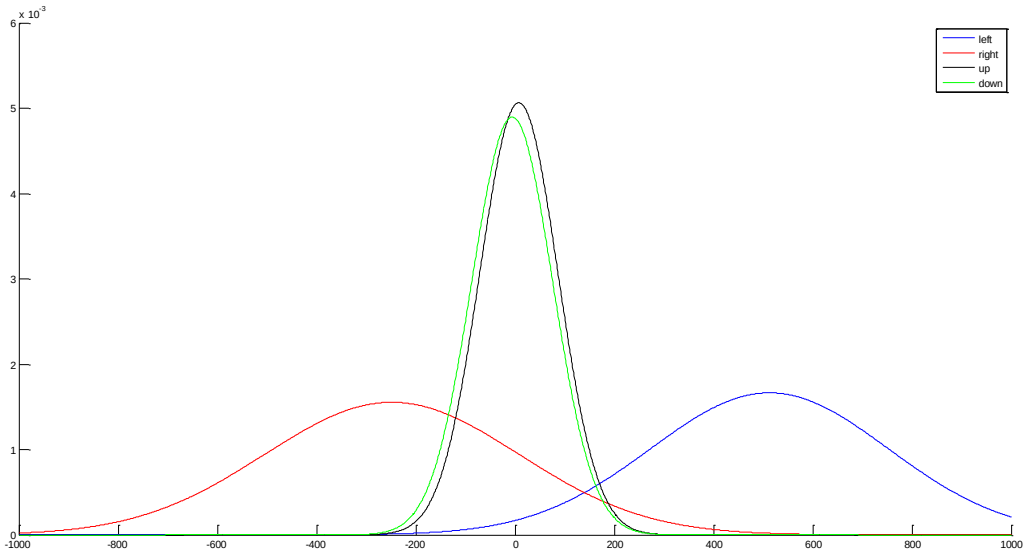

Figura 8. Distribuciones estadísticas de las áreas para las tres clases.

Se puede observar que las áreas calculadas para el movimiento de los ojos hacia la izquierda se distribuyen en valores positivos mientras que para el movimiento hacia la derecha son negativos, y para un movimiento vertical (arriba o abajo) fluctúa entorno a un área muy pequeña en comparación con las direcciones horizontales.

Por tanto, gracias al cálculo de estas áreas, podemos calcular dos umbrales que permitan separar entre las distintas direcciones y así determinar el porcentaje de aciertos y fallos para las distintas clases.

Hasta el momento se ha creado un algoritmo para clasificar las distintas direcciones empleando un método no paramétrico basado en el uso de umbrales, para entender mejor todo este proceso se puede observar en la figura 9 los distintos pasos que se han llevado a cabo para conseguir tal objetivo:

4.1.3 Flujo de trabajo para el algoritmo propuesto.

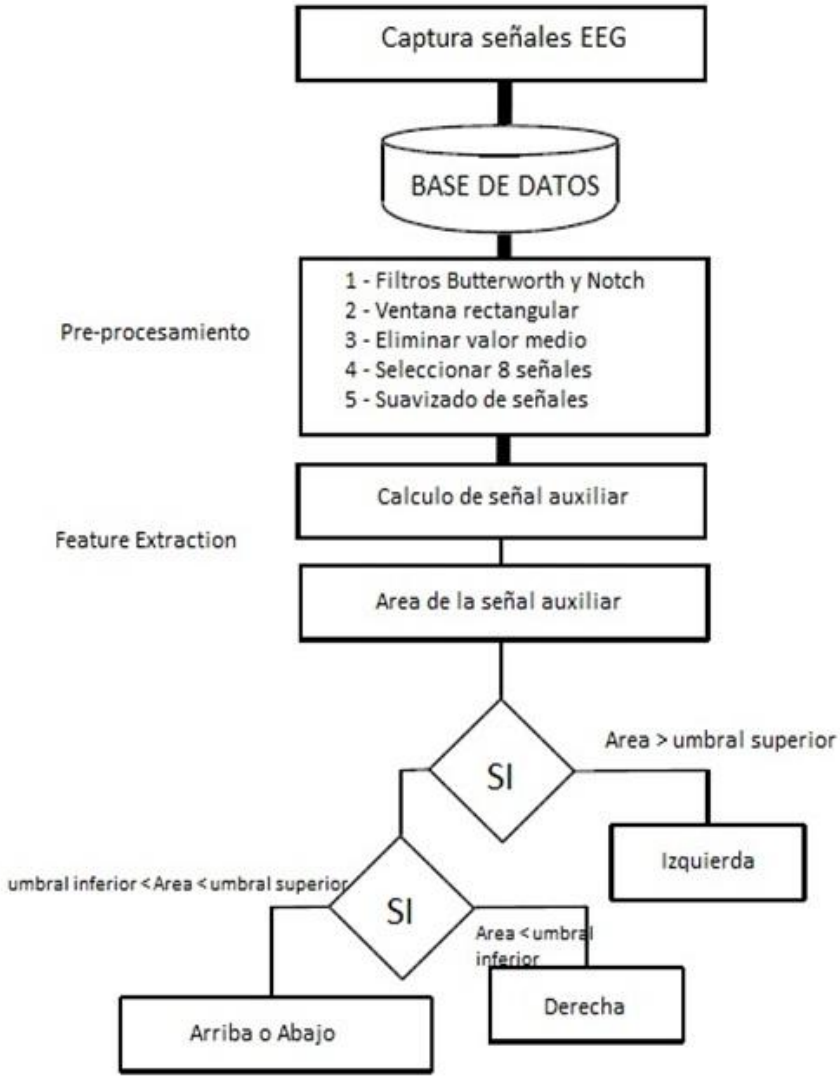

Figura 9. Flujo de trabajo del algoritmo seguido para detectar movimientos oculares usando umbrales.

4.2 Algoritmo de reconocimiento de movimientos oculares basado en el uso de redes neuronales artificiales.

Hasta el momento se ha diseñado un método capaz de clasificar entre distintas direcciones de movimientos oculares basado en el uso de umbrales.

Sin embargo, esta no es la única herramienta que hoy en día se usa con el fin de establecer una clasificación para aplicaciones relacionadas con ondas cerebrales, sino que también se hace uso de múltiples técnicas que utilizan métodos estadísticos para separar entre distintas clases.
Entre las técnicas que se utilizan para estos propósitos destacan el uso de *Redes Bayesanas*, *Modelos Ocultos de Markov* y también se usan distintos algoritmos de Machine Learning como pueden ser el uso de *Support Vector Machine* o *Artificial Neural Network* entre otros.

Para este apartado se ha hecho uso de una red neuronal artificial para la clasificación de las distintas direcciones de movimiento de ojos, en vez de usar un método basado en umbrales como se explicó en el apartado anterior.

Antes que nada, es necesario entender cómo funciona una red neuronal artificial, cuyo objetivo principal es hacer una separación entre distintas clases.
De forma básica, una red neuronal artificial está formada por la interconexión de un grupo de nodos llamados neuronas, las cuales simulan las neuronas biológicas del cerebro de una forma simplificada. Este tipo de redes están estructuradas en tres capas, las cuales son una capa de entrada, una capa oculta y una capa de salida, tal y como se puede visualizar en la figura 10:

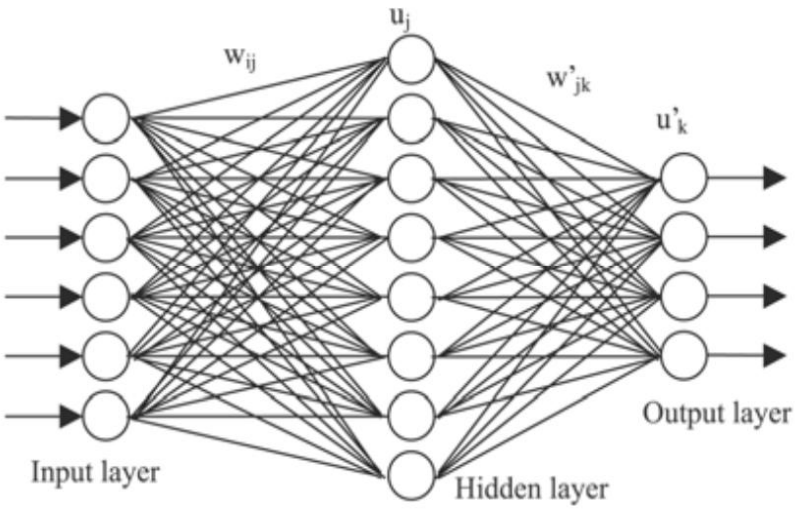

Figura 10. Estructura de una red neuronal artificial.

Como se puede observar en la figura 10, se muestran las tres capas anteriormente dichas y sus correspondientes neuronas, las cuales están representadas mediante círculos, la interconexión de dichas neuronas es lo que da lugar a la red neuronal artificial.

Sin embargo, es necesario también entender de forma clara el funcionamiento de dicha red. Los siguientes apartados explican el funcionamiento principal de una red neuronal artificial.

Una red neuronal artificial consta de dos fases, la primera se conoce como fase de entrenamiento y la segunda recibe el nombre de fase de test o fase de simulación. En los siguientes párrafos se explica de forma detallada como funciona cada una de estas fases:

- La primera fase es conocida como la fase de entrenamiento, y para entenderla es importante saber que una red neuronal aprende gracias a los ejemplos que recibe durante dicha fase. Explicado de una forma sencilla, la red toma como entrada los features que se han extraído de una señal, por lo que si sabemos cuál es la clase a la que pertenecen esos features, se comunica a la red qué neuronas de la capa de salida se deberían de activar. A medida que la red va recibiendo ejemplos para los cuales se conoce la clase correspondiente, y por tanto se conoce también que salidas deberían ser activadas, el propio algoritmo va asignando una serie de pesos para las diferentes neuronas presentes en las tres capas, dando lugar a una compleja función de transferencia. Este proceso es lo que se conoce como la fase de entrenamiento de la red neuronal.

- Como ya se ha comentado más arriba, la segunda fase que poseen este tipo de redes es lo que se conoce como fase de test o fase de simulación.
  En este proceso, gracias a que en la fase de entrenamiento la red neuronal ya tiene los pesos asignados para cada una de las neuronas de sus capas correspondientes, el método a seguir se basa en introducir los features extraídos de los nuevos ejemplos y posteriormente ver que salidas se activan y cuáles no. De esta manera se puede establecer un umbral que permita distinguir entre las clases para las cuales se entrenó la red durante su fase de entrenamiento.

Para este trabajo se entrenó la red neuronal con el objetivo de distinguir entre tres clases: movimiento de ojos hacia la izquierda, derecha y una dirección vertical (arriba o abajo).

El proceso que se ha llevado a cabo para el entrenamiento y test de la red neuronal se explica de forma detallada en los siguientes apartados, así como todos los pasos que han sido necesarios para el pre-procesamiento de las señales EEG y su correspondiente extracción de features.

4.2.1 Pre-procesamiento de las señales EEG.

Al igual que se ha hecho para el algoritmo basado en umbrales, también se ha necesitado realizar un previo procesamiento de las señales cerebrales con el fin de retener sólo las características que forman parte de su naturaleza.
Sin embargo como ya se había mencionado anteriormente, los pasos que se llevan a cabo en el pre-procesamiento de ondas EEG son siempre iguales sea cual sea el propósito final que se quiera conseguir, es por esto que en este segundo método para el cual se ha hecho uso de redes neuronales, se han llevado a cabo los mismos pasos que los realizados en el método basado en el uso de umbrales. Para ello, se explica brevemente a continuación el proceso que se ha llevado a cabo en el pre-procesamiento de las señales.

- Como primer paso es necesario filtrar las señales con un filtro paso-bajo tipo *Butterworth* cuya frecuencia de corte son 60 Hz, y cuyo espectro se encuentra entre 0.5 y 60 Hz, el cual se caracteriza por tener el mayor porcentaje de potencia de señales cerebrales..

- A continuación insertamos un filtro *Notch* para eliminar el ruido que se pueda acoplar procedente de la red eléctrica, el cual puede ser 50 o 60 Hz dependiendo siempre del país. Recordemos que *Emotiv Epoc* suprime por defecto esta frecuencia, aunque es posible que existan interfaces cerebro-máquina las cuales no realicen esta tarea.

- Después del filtrado, se enventana la señal con una ventana tipo rectangular en la zona en la cual se produce un movimiento de ojos, haciéndolo exactamente de la misma manera que se hacía en el algoritmo basado en umbrales.

- El siguiente paso es suprimir el valor medio para así poder tener las señales referenciadas a un valor nulo, el cual permitirá más adelante un mejor tratamiento de los datos.

- Por último se suavizan las señales mediante el mismo método usado anteriormente, es decir aproximamos la señal EEG a un polinomio de orden 10 mediante el método de los mínimos cuadrados, el cual busca los coeficientes del polinomio para el cual se minimiza su función error.

  Cabe recordar que aunque este suavizado se lleva a cabo muchas veces en el pre-procesamiento de señales EEG, el método de los mínimos cuadrados no es el único que se utiliza, existen una gran cantidad de formas de suavizar las señales pero aun así esta técnica funciona de manera razonablemente buena.

De forma muy resumida, estos son los pasos que se han llevado a cabo en la etapa de pre-procesamiento para el método basado en el uso de redes neuronales.
Como ya se había mencionado anteriormente, sea cual sea el propósito final de este tipo de aplicaciones, el procesamiento inicial es siempre el mismo para señales cerebrales, es por esto que tanto para el algoritmo haciendo uso de umbrales como para el algoritmo usando una red neuronal artificial, se llevan a cabo exactamente los mismos pasos.

4.2.2 Extraccion de features.

La extracción de features es un paso necesario e imprescindible a la hora de entrenar una red neuronal artificial.
Es importante saber que una red neuronal aprende gracias al uso de los features que se extraen de señales, para nuestro caso señales cerebrales. Es decir, para que una red neuronal pueda aprender y posteriormente clasificar es necesario pasarle una serie de características que sean discriminantes entre las distintas clases que se quieran clasificar. Por tanto, es necesario buscar features que permitan diferenciar entre clases, aunque como ya se verá más adelante, siempre existirán una gran cantidad tanto de aciertos como de fallos en los resultados.

Uno de los features que se ha extraído para este trabajo ha sido el área bajo la curva de la señal suavizada *X_Auxiliar* en el intervalo en el cual se produce un movimiento de ojos de un sujeto determinado.
Se ha usado principalmente esta área ya que se comprobó al representar las distribuciones estadísticas de los datos, que existía una discriminación suficiente entre las clases, por lo que ha sido también un feature que se ha utilizado para el entrenamiento de la red neuronal artificial.
Al igual que se hizo para el método de los umbrales, también se ha utilizado el método del trapezoide para calcular esta superficie. Esta área se calcula de nuevo con la función *trapz* que proporciona *Matlab*, cuya fórmula matemática se puede visualizar en la ecuación 3.

Otro feature muy usado en el análisis de señales EEG si hablamos del dominio temporal, corresponden a los máximos y mínimos globales de las ondas cerebrales. En este trabajo, los máximos y mínimos varían para las tres clases de forma significativa, por lo que también se decidió usar estos features como entradas para la red neuronal artificial. Para hallar la máxima y mínima amplitud de una señal cualquiera utilizando *Matlab*, solo hay que usar las funciones *max* y *min* respectivamente, las cuales reciben como único parámetro de entrada la señal para la cual queremos hallar estos dos datos, y nos devolverá el máximo y el mínimo de dicha función en el intervalo en el cual está definida.

En resumen, usando solo la señal *X_auxiliar* para la cual se calcula su área bajo la curva, la mínima y máxima amplitud, podemos usar estas tres características como features para entrenar una red neuronal artificial y así posteriormente calcular el porcentaje de aciertos y fallos para las distintas clases que la red ha aprendido a catalogar en su etapa de entrenamiento.

4.2.3 Fase de entrenamiento de la Red Neuronal.

El siguiente paso consiste en entrenar la red neuronal para que pueda aprender a clasificar.

Para este apartado las clases elegidas han sido las mismas que en el método basado en umbrales, dichas clases han sido un movimiento de ojos hacía la derecha, izquierda y una dirección vertical (arriba o abajo), es decir un total de tres clases.

Tanto para el diseño y construcción de la red neuronal como para la fase de entrenamiento y la fase de test se ha utilizado el mismo software. Se ha hecho uso de *Neural Network Toolbox*, el cual es un software de *Matlab* que sirve básicamente para diseñar, construir, entrenar y simular redes neuronales artificiales, incluyendo también una gran cantidad de herramientas y funciones especificas para el manejo de dichas redes.
Por tanto el primer paso que es necesario llevar a cabo es el diseño y la construcción de la red neuronal artificial.

Para empezar su diseño solo hace falta ejecutar *Neural Network Toolbox* en la ventana de comandos de *Matlab*, esto se consigue introduciendo en dicha ventana el comando *nntool* para abrir la interfaz que posee.
Una vez se ha abierto la interfaz el primer paso consiste en importar los datos necesarios que usará la red para la fase de entrenamiento y test, este proceso se consigue mediante el botón *import* que nos ofrece el Toolbox.
Para entender de manera clara todo el procedimiento que se ha seguido, a partir de ahora en este trabajo se nombran una serie de vectores los cuales tienen su correspondiente función:

- *Input*: Vector correspondiente a los features que se usan para la fase de entrenamiento de la red neuronal. Los elementos de este vector contienen los tres features explicados anteriormente: el área bajo la curva, la máxima y mínima amplitud de la señal *X_auxiliar*.

- *Target*: Vector correspondiente a las salidas deseadas de la red neuronal para los datos de *input*. Es decir, es un vector que especifica qué neuronas de la capa de salida se deben de activar de acuerdo a los datos de entrada durante su fase de entrenamiento.

- *Sample*: Vector que contiene los mismos features que el vector *input*, pero en este caso son los correspondientes a la fase de test.

Una vez importados estos tres vectores, se puede ver en la figura 11 el formato que debería de tener la interfaz para asegurar que han sido correctamente importados.

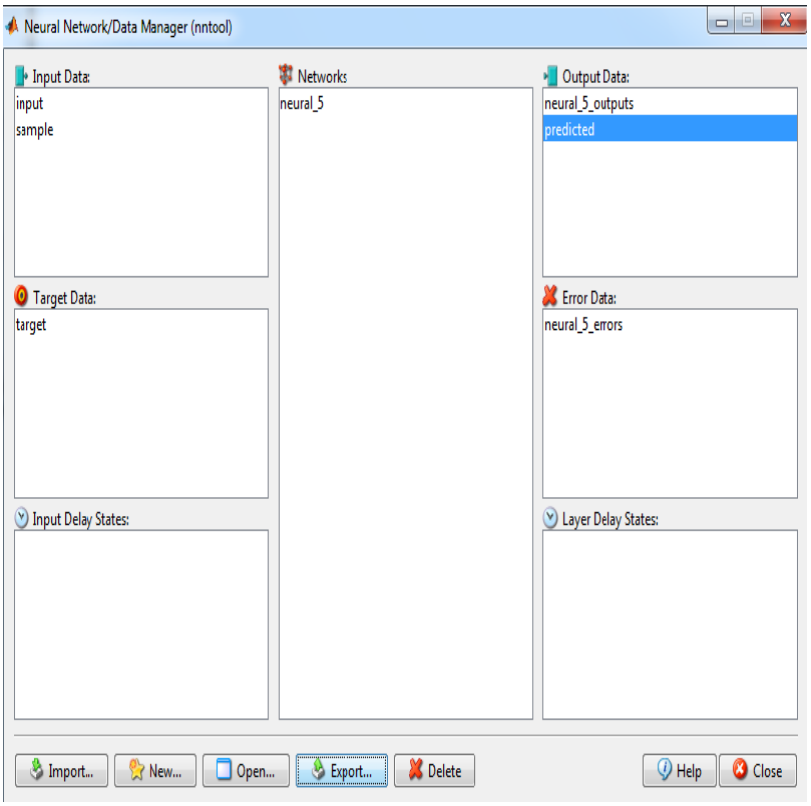

Figura 11. Captura de pantalla 1 de Neural Network Toolbox.

En la figura 11 se puede ver que tanto *input* como *sample*, al ser vectores que contienen features, estos necesitan ser entradas a la red neuronal, por lo que es necesario que se sitúen en el campo *Input Data*, mientras que *target* necesita estar en el campo *Target Data*. Este paso se necesita realizar para comunicar a la red neuronal cuales van a ser los vectores que van a ser considerados como entradas a la red y cuales serán considerados como datos target.

El siguiente paso consiste en el diseño y construcción de la red neuronal.
Para esto, solo hay que crear una nueva red mediante la opción *New* (ver figura 11).
En la siguiente ventana que se abre una vez pulsada esta opción, es necesario introducir los vectores que se usarán para entrenar la red neuronal, así como una serie de parámetros que la red utilizará para su diseño. Entre los parámetros importantes destacan el tipo de red, el cual en este trabajo será una red tipo *Feed-forward-backpropagation*.

También ha sido necesario introducir el número de capas que queremos que tenga la red neuronal, en este caso se ha hecho uso de 2 capas, y posteriormente se le indica el número de neuronas necesarias en la capa oculta y en la capa de salida. En este trabajo se han usado 14 neuronas para la capa oculta y 3 para la capa de salida. También es necesario seleccionar la función de transferencia que usará.

Una vez rellenados todos estos parámetros, finalizamos la creación de dicha red mediante la opción *Create*, la cual si están rellenados todos los datos correctamente mostrará un mensaje de confirmación.
Después de finalizar este diseño, se puede comprobar de una manera gráfica la estructura que posee la red neuronal.

La figura 12 muestra un dibujo sobre el formato que tiene la red neuronal una vez ha sido diseñada:

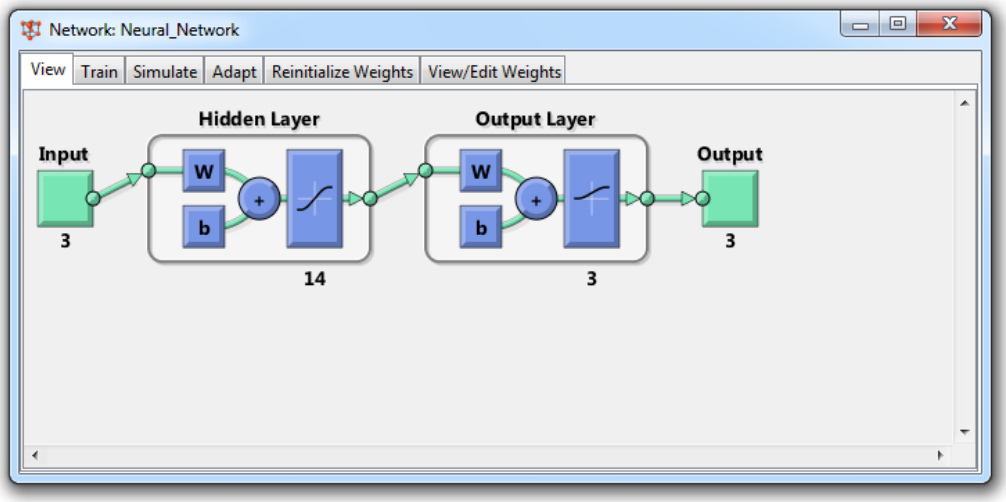

Figura 12. Captura de pantalla 2 de Neural Network Toolbox.

A continuación, el siguiente paso a realizar es el entrenamiento de la red. Para llevar esto a cabo, tal y como se puede visualizar en la figura 13, mediante la opción *Train* podemos insertar los parámetros necesarios para su entrenamiento:

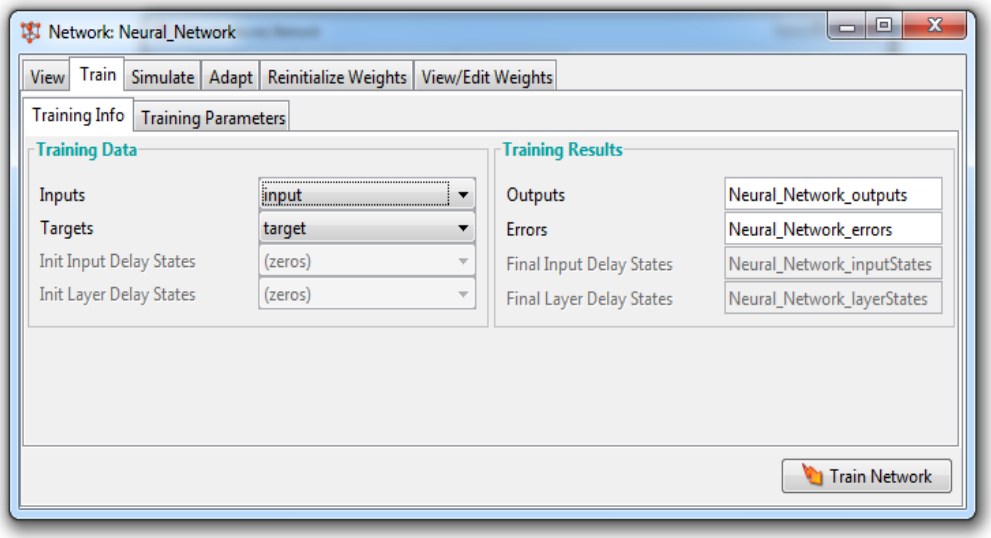

Figura 13. Captura de pantalla 3 de Neural Network Toolbox.

Estableciendo los parámetros tal y como indica la figura 13 podemos posteriormente entrenar la red gracias a la opción *Train Network.* Una vez entrenado aparecerá una ventana con mensaje de confirmación informando de que dicha red ha sido entrenada de forma correcta.
La figura 14 muestra el formato que tiene dicha ventana una vez todo haya sido procesado de forma correcta:

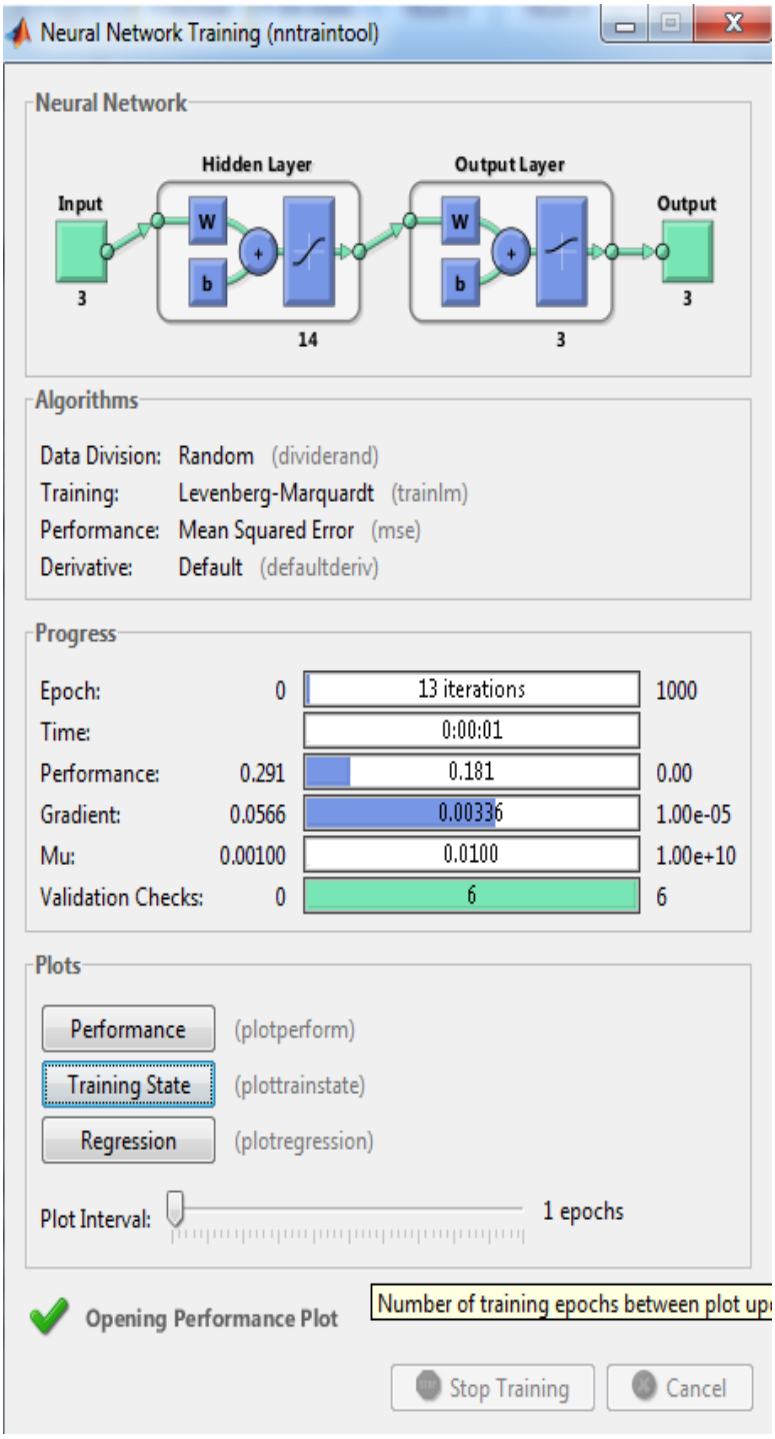

Figura 14. Captura de pantalla 4 de Neural Network Toolbox.

4.2.4 Fase de Test de la red neuronal artificial.

El segundo paso que se lleva a cabo siempre en una red neuronal consiste en la fase de simulación o fase de test. Este proceso consiste en la simulación de dicha red usando muestras diferentes a las usadas en la fase de entrenamiento, y mediante estas nuevas muestras la red neuronal activará las salidas correspondientes, siempre de acuerdo a como haya aprendido la red en la fase de entrenamiento.

Para esta fase la manera por la que podemos simular nuestra red es mediante la opción *Simulate* (ver figura 15). En este paso solo hay que asignar al campo *Inputs* el vector de entrada que se quiere utilizar para realizar dicha simulación. Como ya se mencionó anteriormente, los datos que se han utilizado para la simulación de la red corresponde al vector *sample*.
Una vez introducidos dichos datos, el formato que debe tener es el que muestra la figura 15:

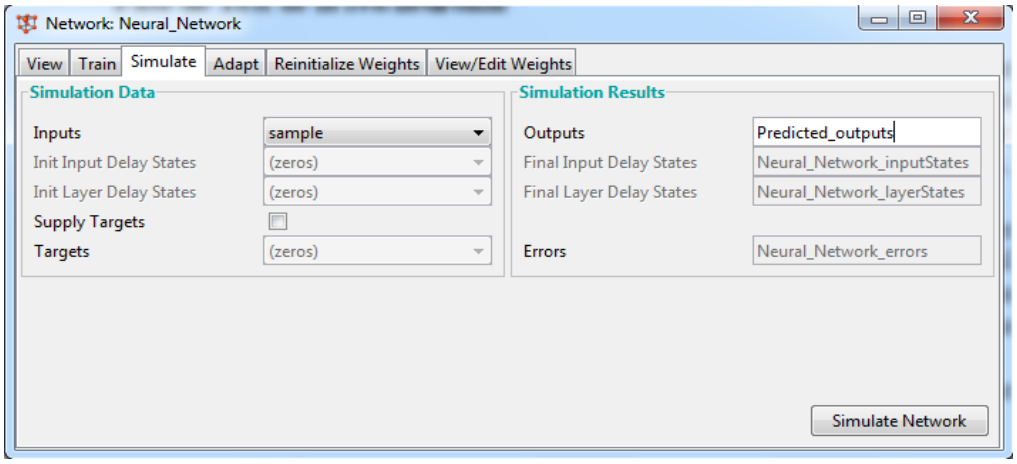

Figura 15. Captura de pantalla 5 de Neural Network Toolbox.

Para acabar, solo hace falta pulsar la opción *Simulate Network*, este proceso finalmente calculará las salidas de la red neuronal para los datos usados en la fase de test, es decir, calculará las salidas cuando las entradas corresponden a los features que contiene el vector *sample*. Dichas salidas se almacenarán en el vector *Predicted_outputs* (ver figura 15).

Por tanto, este proceso contiene todos los pasos necesarios para la simulación de la red neuronal artificial.

4.2.5 Flujo de trabajo para el algoritmo propuesto.

De la misma forma que se hizo en el algoritmo basado en umbrales, también se puede ver a continuación en la figura 16 el flujo de trabajo seguido para el algoritmo utilizando una red neuronal artificial:

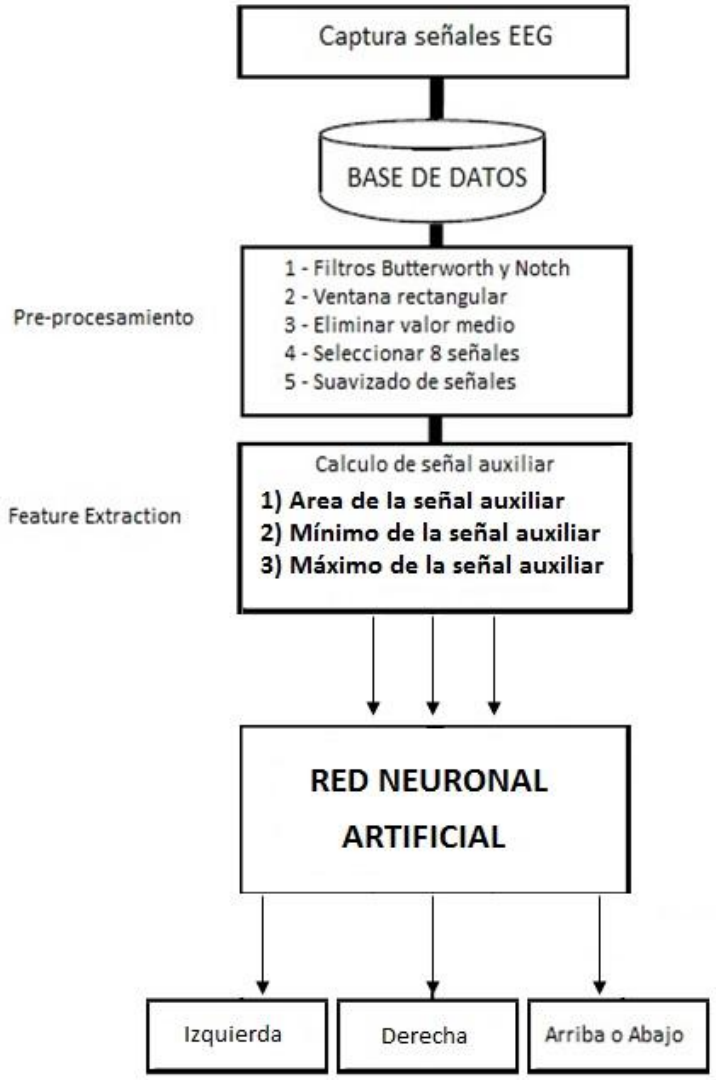

Figura 16. Flujo de trabajo del algoritmo seguido para detectar movimientos oculares usando una red neuronal artificial.

Una vez hecho todo este proceso, se procedió a calcular el porcentaje de aciertos y fallos para las tres clases y para cada uno de los usuarios a los que se realizó el experimento. El método para calcular estas estadísticas consiste en establecer un umbral a partir del cual se considera que una salida está activada.
En la fase de entrenamiento de la red neuronal, cada salida de dicha red tiene asociada una determinada clase, por lo que si una salida se activa entonces se considera que la

red ha reconocido su clase correspondiente, mientras que si no se activa o se activan varias salidas a la vez entonces no se ha reconocido esa clase o ninguna respectivamente.
En este trabajo se entrenó la red neuronal para activar las salidas con un valor igual a 1 y desactivarlas con un valor igual a 0.

Sin embargo, las redes neuronales no son perfectas por lo que en la fase de simulación la red neuronal intentará activar sus salidas con valores cercanos a los que se utilizó para entrenar la red, en nuestro devolverá salidas cuyos valores sean cercanos a 1 o 0. Es aquí donde es necesario hallar un umbral a partir del cual se considera que una salida está activada o desactivada, detectando en consecuencia si una clase ha sido reconocida o no.

5 Integración, pruebas y resultados

En esta sección se muestran los resultados finales que se han obtenido para el algoritmo de reconocimiento de movimientos oculares, tanto para el método basado en el uso de umbrales como el método basado en el uso de una red neuronal artificial.

Para ello, se pueden visualizar en distintas tablas, el porcentaje de aciertos y fallos para ambos algoritmos y para los 5 sujetos a los que se realizó el experimento.

Por último, se muestra una comparativa de ambos algoritmos, mostrando el porcentaje de aciertos y fallos realizando una media global de todos los usuarios. De esta manera se puede tener una visión global sobre ambos sistemas, para así poder tener una idea de qué sistema es más eficiente para los algoritmos diseñados en este trabajo.

5.1 Resultados para el algoritmo basado en el uso de umbrales.

En este apartado se pueden visualizar en un conjunto de tablas, el porcentaje de aciertos y fallos para las distintas direcciones de movimiento de ojos para cada uno de los usuarios a los que se realizó el experimento, mediante el método basado en el uso de umbrales.

Para calcular estas estadísticas, se diseñó un código en *Matlab* que permitiese calcular el porcentaje de aciertos y fallos para las distintas clases, haciendo uso de un contador que se incrementase cuando se acertase con la clase correspondiente.

Las señales para las que se han calculado estos datos corresponden a las 25 muestras restantes de la base de datos para cada uno de los usuarios. Es decir, se usaron 25 muestras para el cálculo de los umbrales, y las 25 restantes se han utilizado para determinar las estadísticas mencionadas.

Las tablas en las que se puede visualizar el porcentaje de aciertos y fallos corresponden a las tablas 1 a 5.

Sujeto 1	**Aciertos**	**Fallos**
Dirección izquierda	48 %	52 %
Dirección derecha	40 %	60 %
Dirección arriba/abajo	14 %	86 %

Tabla 1. Aciertos y Fallos - Sujeto 1 – Algoritmo basado en umbrales.

Sujeto 2	Aciertos	Fallos
Dirección izquierda	68 %	32 %
Dirección derecha	68 %	32 %
Dirección arriba/abajo	96 %	4 %

Tabla 2. Aciertos y Fallos - Sujeto 2 – Algoritmo basado en umbrales.

Sujeto 3	Aciertos	Fallos
Dirección izquierda	64 %	36 %
Dirección derecha	56 %	44 %
Dirección arriba/abajo	96 %	4 %

Tabla 3. Aciertos y Fallos - Sujeto 3 – Algoritmo basado en umbrales.

Sujeto 4	Aciertos	Fallos
Dirección izquierda	48 %	52 %
Dirección derecha	28 %	72 %
Dirección arriba/abajo	30 %	70 %

Tabla 4. Aciertos y Fallos - Sujeto 4 – Algoritmo basado en umbrales.

Sujeto 5	Aciertos	Fallos
Dirección izquierda	4 %	96 %
Dirección derecha	28 %	72 %
Dirección arriba/abajo	94 %	6 %

Tabla 5. Aciertos y Fallos - Sujeto 5 – Algoritmo basado en umbrales.

Una de las principales características que se puede concluir en relación a las tablas 1 a 5, es que el porcentaje de aciertos y fallos puede variar considerablemente de un usuario a otro, por ejemplo para el usuario número 5 el porcentaje de aciertos para la dirección hacía la izquierda es del 4 % mientras que para el usuario 2 es del 68 %.

La principal dificultad que se ha encontrado en este trabajo ha sido encontrar un feature que permita clasificar entre la dirección hacia arriba y hacia abajo. Principalmente esto es debido a que en estos casos, las señales no mostraban variaciones significantes entre sí, por lo que no se ha podido encontrar ninguna forma de distinguir entre las dos direcciones verticales.

5.2 Resultados obtenidos para el algoritmo basado en el uso de una red neuronal artificial.

En este apartado, al igual que se hizo para el método basado en el uso de umbrales, se han calculado el porcentaje de aciertos y fallos para cada uno de los distintos sujetos:

Las tablas 6 a 10 corresponden a las estadísticas anteriormente mencionadas:

Sujeto 1	**Aciertos**	**Fallos**
Dirección izquierda	12 %	88 %
Dirección derecha	4 %	96 %
Dirección arriba/abajo	0 %	100 %

Tabla 6. Aciertos y fallos – Sujeto 1 – Algoritmo basado en redes neuronales.

Sujeto 2	**Aciertos**	**Fallos**
Dirección izquierda	96 %	4 %
Dirección derecha	84 %	16 %
Dirección arriba/abajo	100 %	0 %

Tabla 7. Aciertos y fallos – Sujeto 2 – Algoritmo basado en redes neuronales.

Sujeto 3	Aciertos	Fallos
Dirección izquierda	68 %	32 %
Dirección derecha	84 %	16 %
Dirección arriba/abajo	100 %	0 %

Tabla 8. Aciertos y fallos – Sujeto 3 – Algoritmo basado en redes neuronales.

Sujeto 4	Aciertos	Fallos
Dirección izquierda	76 %	24 %
Dirección derecha	36 %	64 %
Dirección arriba/abajo	78 %	22 %

Tabla 9. Aciertos y fallos – Sujeto 4 – Algoritmo basado en redes neuronales.

Sujeto 5	Aciertos	Fallos
Dirección izquierda	80 %	20 %
Dirección derecha	84 %	16 %
Dirección arriba/abajo	78 %	22 %

Tabla 10. Aciertos y fallos – Sujeto 5 – Algoritmo basado en redes neuronales.

A la vista de los resultados obtenidos, se puede concluir que para la mayoría de los usuarios la red neuronal funciona de una manera eficiente con un porcentaje de aciertos elevado, usando tan solo los tres features correspondientes al área, máxima amplitud y mínima amplitud de la señal suavizada *X_Auxiliar*.

Como se puede comprobar, este es otro método que hace uso de una red neuronal artificial, en vez de un algoritmo basado en la utilización de umbrales para diferenciar entre distintas clases.

5.3 Comparativa de ambos algoritmos.

Este apartado es una sección importante para este trabajo, ya que es un apartado en el cual se muestran dos tablas, cada una de las cuales muestra en media para los 5 usuarios escogidos para el experimento, el porcentaje de aciertos y fallos para las 3 clases del experimento para ambos algoritmos usados.

Gracias al cálculo de dichas tablas, podemos comparar el porcentaje de aciertos y fallos para cada uno de los métodos usados y así poder tener una visión global de ambos.

Las tablas 11 y 12 muestran, en media, el porcentaje de aciertos y fallos para ambos algoritmos desarrollados.

La tabla 11 corresponde al algoritmo basado en el uso de umbrales mientras que la tabla 12 corresponde al algoritmo usando una red neuronal artificial:

Porcentajes medios para 5 sujetos	**Aciertos**	**Fallos**
Dirección izquierda	46.4 %	53.6 %
Dirección derecha	44 %	56 %
Dirección arriba/abajo	66 %	34 %

Tabla 11. Aciertos y fallos – Valores medios – Algoritmo basado en umbrales.

Porcentajes medios para 5 sujetos	**Aciertos**	**Fallos**
Dirección izquierda	66.4 %	33.6 %
Dirección derecha	58.4 %	41.6 %
Dirección arriba/abajo	71.2 %	28.8 %

Tabla 12. Aciertos y fallos – Valores medios – Algoritmo basado en redes neuronales.

A la vista de las tablas obtenidas, podemos concluir que el algoritmo desarrollado usando redes neuronales artificiales para clasificar las tres clases funciona, en media, mejor que el caso que utiliza una técnica no paramétrica basada en umbrales.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo se han desarrollado dos algoritmos distintos para identificar y clasificar movimientos de ojos, a partir de las señales cerebrales que proporciona la interfaz cerebro-máquina *Emotiv Epoc*.

Estos dos algoritmos funcionan con una tasa razonable de aciertos como se ha visto en las tablas anteriores, aunque como ya se ha podido comprobar, pueden existir usuarios para los cuales estas estadísticas son bastante malas en comparación con el resto, aunque en media funcione de manera razonable.

Una de las principales conclusiones a las que se ha podido llegar ha sido que para este tipo de señales, existen multitud de técnicas para clasificar tareas mentales, o movimientos de ojos entre otros.

También una de las principales características que poseen señales de esta naturaleza se basa en el hecho de que proporcionan información relacionada con funciones cerebrales, pero siempre de una manera indirecta, por lo que como ya se ha explicado es necesario un profundo procesamiento de las señales y posterior análisis para poder extraer la información que estas proveen.

Hoy en día, aun es necesario mucha más investigación relacionada con métodos y algoritmos que permitan entender mejor este tipo de señales, para así poder desarrollar tecnologías que faciliten en gran medida la vida a gente que posean discapacidades motoras o del habla.

6.2 Trabajo futuro

Para este trabajo se han utilizado las señales EEG para obtener información relativa a movimientos oculares, sin embargo las interfaces cerebro-máquina obtienen como ya hemos mencionado información que tiene que ver con actividades mentales.

Un posible trabajo futuro que se podría desarrollar sería crear un algoritmo que a partir de señales EEG, permita clasificar distintas actividades mentales, como por ejemplo, clasificar si un sujeto está pensando en mover el brazo derecho o el brazo izquierdo.
Para conseguir esta clasificación, las técnicas que se suelen usar son métodos que involucran el dominio de la frecuencia como análisis principal, como por ejemplo el cálculo de la densidad espectral de potencia (PSD) de dos electrodos opuestos.

Referencias

[1] Paulo Andrés Vélez Á, Haeders Saldarriaga V, Humberto Loaiza C, "*Técnicas de clasificación para neuroseñales*".

[2] Abdelkader Nasreddine Belkacem, Hideaki Hirose, Natsue Yoshimura, Duk Shin, Yasuharu Koike, "*Classification of Four Eye Directions from EEG Signals for Eye-Movement-Based Communication Systems".*

[3] Germán Rodríguez Bermúdez, Pedro J. García Laencina, Domitien Brizion, Joaquín Roca Dorda, *"Adquisición, procesamiento y clasificación de señales EEG para el diseño de sistemas BCI basados en imaginación de movimiento".*

[4] Shirley Cordova Villar, Willian A. Perez Oviedo, Avid Román Gonzalez, *"Implementación de métodos de procesamiento de señales EEG para aplicaciones de comunicación y control".*

[5] Abdul-Bary Raouf Suleiman, Toka Abdul-Hameed Fatehi, *"Features extraction techniques of eeg signal for BCI applications".*

[6] Nandish.M, Stafford Michahial, Hemanth Kumar P, Faizan Ahmed, *"Feature extraction and classification of EEG signal using neural network based techniques".*

[7] Hiran Ekanayake, *"P300 and Emotiv EPOC: Does Emotiv EPOC capture real EEG?"*

[8] Brijil Chambayil, Rajesh Singla, R. Jha, *"EEG Eye Blink Classification Using Neural Network".*

Anexos

Anexo A

Código necesario para el pre-procesamiento de señales EEG:

```
% Devuelve las señales EEG filtradas y suavizadas en la ventana
elegida
function [eeg_result]=eeg_process(datos)


% Primero necesitamos aplicar un filtro Butterworth (de 0.5 a 60 Hz
para coger solo la señal EEG)
orden=8;                    % Butterworth order filter
Fs=128;                     % Frecuencia de muestreo del BCI
frec_corte=60;
Wn=frec_corte/(Fs/2);       % Normalized cut-off frequency
[b,a] = butter(orden,Wn);

for i=3:16
    datos_filtrados(i,:)=filter(b,a,datos(i,:));
end


% Aplicar tambien filtro notch para eliminar el ruido de la red
electrica (a 50 Hz en España).
frec_notch=50;
Wo=frec_notch/(Fs/2);
Q=35;
BW=Wo/Q;
[NUM,DEN] = iirnotch(Wo,BW);
for i=3:16
    datos_filtrados(i,:)=filter(NUM,DEN,datos_filtrados(i,:));
end

% Select rectangular window in the area where the eyes are moving
rect_window=[768:768+128];
datos_window=datos(:,rect_window);


% Remove baseline
baseline=[];
for i=1:16
    baseline(i)=mean(datos_window(i,:));
end;

for i=3:16
    for j=1:length(datos_window)
        datos_window(i,j)=datos_window(i,j)-baseline(i);
    end;
end;
```

```
% Las señales AF3/AF4 F3/F4 F7/F8 Y FC5/FC6 son las que mas
informacion dan acerca del movimiento de ojos


x=[1:1:length(datos_window)];
orden=10;


% Primero elimino ruido y suavizo la señal, mediante LEAST-SQUARE
METHOD,con orden a determinar.
%p = polyfit(x,y,n) returns the coefficients for a polynomial p(x) of
degree n that is a best fit (in a least-squares sense) for the data in
y. The coefficients in p are in descending powers, and the length of p
is n+1

p3=polyfit(x,datos_window(3,:),orden);
p4=polyfit(x,datos_window(4,:),orden);
p5=polyfit(x,datos_window(5,:),orden);
p6=polyfit(x,datos_window(6,:),orden);

p16=polyfit(x,datos_window(16,:),orden);
p15=polyfit(x,datos_window(15,:),orden);
p14=polyfit(x,datos_window(14,:),orden);
p13=polyfit(x,datos_window(13,:),orden);


intervalo=1:length(datos_window)

polinomio_p3=polyval(p3,intervalo)
polinomio_p4=polyval(p4,intervalo)
polinomio_p5=polyval(p5,intervalo)
polinomio_p6=polyval(p6,intervalo)

polinomio_p16=polyval(p16,intervalo)
polinomio_p15=polyval(p15,intervalo)
polinomio_p14=polyval(p14,intervalo)
polinomio_p13=polyval(p13,intervalo)

eeg_result=[polinomio_p3;polinomio_p4;polinomio_p5;polinomio_p6;polino
mio_p13;polinomio_p14;polinomio_p15;polinomio_p16];
```